2014 47th Annual IEEE/ACM International Symposium on Microarchitecture

Citadel: Efficiently Protecting Stacked Memory from Large Granularity Failures

Prashant J. Nair† David A. Roberts‡ Moinuddin K. Qureshi†

†*School of Electrical and Computer Engineering*
Georgia Institute of Technology
{pnair6,moin}@ece.gatech.edu

‡*AMD Research*
Advanced Micro Devices Inc.
david.roberts@amd.com

***Abstract*—Stacked memory modules are likely to be tightly integrated with the processor. It is vital that these memory modules operate reliably, as memory failure can require the replacement of the entire socket. To make matters worse, stacked memory designs are susceptible to newer failure modes (for example, due to faulty through-silicon vias, or TSVs) that can cause large portions of memory, such as a bank, to become faulty. To avoid data loss from large-granularity failures, the memory system may use symbol-based codes that stripe the data for a cache line across several banks (or channels). Unfortunately, such data-striping reduces memory level parallelism causing significant slowdown and higher power consumption.**

This paper proposes *Citadel*, a robust memory architecture that allows the memory system to retain each cache line within one bank, thus allowing high performance, lower power and efficiently protects the stacked memory from large-granularity failures. Citadel consists of three components; *TSV-Swap*, which can tolerate both faulty data-TSVs and faulty address-TSVs; Tri Dimensional Parity (3DP), which can tolerate column failures, row failures, and bank failures; and *Dynamic Dual-Granularity Sparing (DDS)*, which can mitigate permanent faults by dynamically sparing faulty memory regions either at a row granularity or at a bank granularity. Our evaluations with real-world data for DRAM failures show that Citadel provides performance and power similar to maintaining the entire cache line in the same bank, and yet provides 700x higher reliability than ChipKill-like ECC codes.

***Keywords*-DRAM, Stacked Memory, Faults, Through Silicon Vias, Error Correcting Code, Resilience**

I. Introduction

The emerging 3D stacked DRAM technology can help with the challenges of power consumption, bandwidth demands and reduced footprint. One of the key enablers of stacked memory is the *through-silicon via* (TSV) technology, which makes it possible to cost-effectively stack multiple memory dies on top of each other [1]. The shorter internal data paths afforded by TSVs reduce capacitance and active power. By exploiting wide buses [2] or high-frequency SerDes interfaces [3] and higher levels of internal parallelism, both bandwidth and random-access latency are improved. It is anticipated that high-performance stacked memories often will be permanently attached to host processors via direct stacking, silicon interposers or other hard-wired interconnects. In such a system, memories that develop permanent faults must continue to work, in order to avoid replacement of multiple chips when tends to be expensive. These factors motivate the adoption of a *fail-in-place* philosophy for designing stacked memory systems [4].

Recent work on DRAM reliability [5] showed that large-granularity DRAM chip failures, such as bank failures, occur nearly as frequently as single-bit failures in commodity DIMMs. Stacked memory designs would not only be subjected to these failures but also to newer fault models, such as arising from faulty TSVs, which can cause failures of several dies, or column failures or bank failures. Thus, stacked memory systems will be significantly more vulnerable to large-granularity failures. Unfortunately, conventional error correction schemes such as ECC DIMMs [6] are targeted towards correcting random bit errors and are ineffective at tolerating large-granularity faults. Memory systems can tolerate large granularity failures using symbol-based coding schemes like ChipKill [7]. However, this increases the number of activated chips and total power consumption.

To optimize performance and power for stacked memory, we want to retain the data for a cache line within a single bank. However, a bank failure would then cause loss of data for the whole cache line. One can adopt a philosophy similar to ChipKill for tolerating large-granularity failures for stacked DRAM. In such a design, the data for a cache line would be striped across several banks (or channels), and a symbol-based coding can be applied, in which the size of each symbol would be equal to the amount of data stored in each bank. Unfortunately, such a data mapping would require the memory system to activate several banks to service a single request. This causes performance degradation (10% to 25%) due to loss of bank(channel) level parallelism, and power consumption (as high as 6x in our evaluations) due to activation of several banks to service one request.

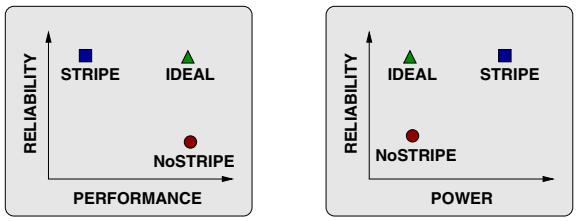

Figure 1. Striping enhances reliability but sacrifices performance and power efficiency. Ideally, we want to tolerate large-granularity failures at high performance and low power.

As shown in Figure 1, ideally we want a system that has the performance and power efficiency of storing the entire cache line in one bank (NoStripe), and yet maintains robustness to large granularity faults (Stripe). To that end, this paper proposes *Citadel*, a robust memory architecture that allows the memory system to retain each cache line within one bank (delivering high performance and low power) and yet efficiently protects the stacked memory from large-granularity failures.

1072-4451/14 $31.00 © 2014 IEEE
DOI 10.1109/MICRO.2014.57

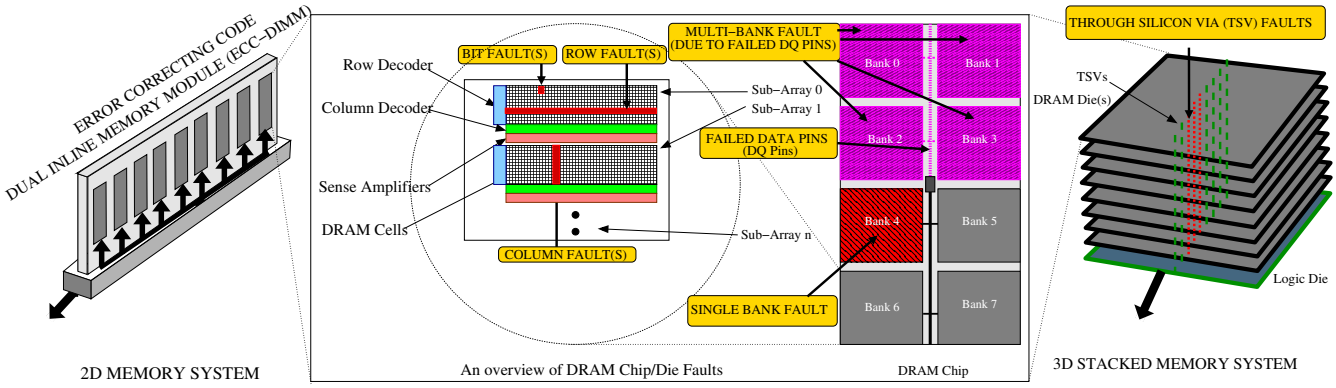

Figure 2. Granularity of faults that occur in a DRAM Chip/Die. Faults can be at granularities of bit, column, row, bank(s), TSVs and I/O links for stacked memory systems. Common wiring faults within a chip can cause multiple banks to fail.

Like ECC DIMMs which have one additional chip per 8 chips, in our study, Citadel has one extra die ECC die with smaller rows along with eight data dies. Similar to an ECC-DIMM that provides 64 bits of ECC every 512 bits, Citadel uses the 64 bits of metadata associated with each 512-bit cache line. Based on key insights, Citadel employs three-pronged approach for fault tolerance.

Insight 1- Protect Against Runtime TSV Faults: As faulty TSVs are a major cause of multi-bank failures in stacked memory, our first idea, *TSV-Swap*, specifically targets TSV faults that happen at runtime. Contrary to manufacture-level spare TSVs [8], that are used to repair faulty TSVs at design time, TSV-SWAP does not rely on any manufacturer-provided spare TSVs. Instead, TSV-Swap dynamically exchanges faulty TSVs with non-faulty TSVs with a remapping circuit. We found that while a data TSV typically affects only one bit in a data line (albeit across many lines), a failure of one of the address TSVs can make half of the memory unreachable. Thus, address TSVs are much more critical than data TSVs for system reliability. Our proposed implementation of TSV-Swap can repair upto 8 faulty TSVs which can be data, address or command TSVs.

Insight 2- Detect and Correct Large Granularity Failures: Even after mitigation of TSV related faults, the stacked memory is still vulnerable to internal DRAM die faults. We want to protect stacked memory not only from small granulity failures (such as bit fault or word fault) but also from large granularity faults such as column-fault, row-faults or even complete bank failures. Our second idea, *Tri Dimensional Parity (3DP)*, provides highly effective and storage efficient correction for both small and large granularity failures. The 3DP proposal maintains parity in three dimensions: 1) Across all banks and dies for individual rows. 2) Across all rows in all banks within a die. 3) Across all rows in single bank across all dies. Each line is equipped with CRC-32 [9] to detect data errors. If any error is detected, it is corrected using the parity information of 3DP. 3DP design provides 130x higher resilience than just applying 2D-ECC. To achieve this, it incurs only 1.6% storage overhead, compared to the 25% storage required for the prior 2D schemes.

Insight 3- Isolate Faulty Memories with Efficient Sparing: When a fault is detected, data is restored using the correction capability of 3DP. However, modules with permanent faults would incur the correction overheads frequently. To avoid such frequent correction, we would like to redirect a faulty memory unit to a spare area. Unfortunately, if the sparing granularity is too fine, then it incurs significant tracking overheads (for example, if a bank fails then thousands of rows get spared to the spare area). If the sparing granularity is too coarse the it results in significant wasted space (for example, sparing at a bank granularity would be wasteful if only one row is faulty). We make a key observation that a bank typically has either one or two row failures, or has thousands of row failures (due to a sub-array or bank failure). Our third idea, *Dynamic Dual-Grained Sparing (DDS)*, exploits the bimodal behavior of faulty units and efficiently spares either at a row or bank granularity. Our proposed design of DDS can spare two faulty banks along with several row failures.

We perform reliability studies using real field data and perform sensitivity studies when field data is unavailable (e.g. for TSVs). Our evaluations, with an industry-grade fault simulator [10], shows that Citadel provides 100x-1000x higher reliability while still retaining power and performance similar to a system that maps the entire cache line in the same bank. To achieve this, Citadel requires a storage overhead similar to that of ECC DIMMs (14% vs. 12.5%).

II. Background and Motivation

Stacked memory systems have lower energy per bit and higher bandwidth when compared to their 2D counterparts. However, to obtain the power-efficiency and high bandwidth of stacked memory, the system must first address reliability challenges. As shown in Figure 2, failures can occur in a memory system at different granularities [5, 11, 12, 13].

A. Memory Faults for Traditional Systems

A memory DIMM consists of multiple DRAM chips. A DRAM chip is organized into banks, where all banks share a common data bus. These banks are composed of rows and columns and are divided into sub-arrays. The banks contain row and column decoders that activate the wordlines or select bitlines associated with the memory request. Faults at the DIMM level can affect all DRAM chips within a DIMM. However, the faults in individual chips are largely independent of each other. In this paper, the definitions for the chip faults follow that of Sridharan et. al. [5] and are represented in Figure 2. We note that banks are operated almost independently and share only wiring such as data, address and command buses [14, 15]. Bank and rank faults occur mainly from faulty data or address or command buses.

B. Transposing Faults onto 3D Stacked Memories

Layout of an individual die in 3D stacked memory systems shows that its internal organization is very similar to that of a chip in conventional 2D memory systems [16, 17, 18, 19]. To a first order, this paper transposes failure rates for all fault types except complete bank and complete rank for current 2D memory system onto stacked memory systems. The key difference is the introduction of TSVs for connecting data and address lines [1]. Due to this, complete bank faults and complete rank faults in any 3D stacked memory are now influenced by TSV faults.

C. Stacked Memory: Organization and ECC Layout

There are several design prototypes of stacked memory, including the High Bandwidth Memory (HBM) [3], Hybrid Memory Cube (HMC) [2, 17] and Octopus from Tezzaron [20]. These standards differ in their data organization and also share TSVs differently. However, these stacked memory systems fundamentally have the same layout. In our study, we analyze an HBM like design (however we found that the reliability improvement with our proposal is equally high for the HMC and Tezzaron designs). Figure 3 shows internal stack organizations of HBM. Each channel may be fully contained in each DRAM die in the stack. A complete set of TSVs and buffers connect each channel to the external interface.

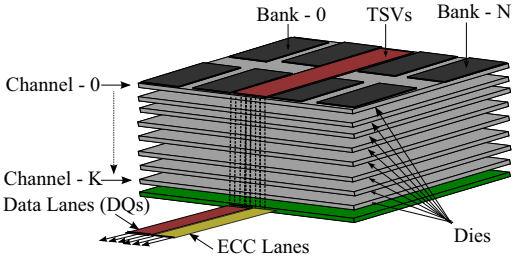

Figure 3. High Bandwidth Memory has a channel(s) per die and all banks in this channel are on the same die. HBM specification includes separate Data and ECC lanes

The stacked memory consists of D data dies and E ECC dies (depending on value of D and the ECC implementation). ECC can be stored in an additional space provided by D dies or can be distributed across $D + E$ dies. Similar to ECC-DIMMs, every data request for a 512b data line also concurrently fetches its 64b ECC metadata through dedicated ECC lanes [3]. In our paper, we use an 8-die stack with one additional ECC die for ECC or metadata information. Our organization has the same storage overhead as incurred in ECC DIMMs (12.5%).

D. Data Striping in 3D Memory Systems

The way data is striped in the memory system has a significant impact not only on power and performance but also reliability of the overall system. A conventional (2D) DIMM stripes a cache line across several chips. Similarly, a stacked memory system can place the cache line in one of three ways:

- **Same Bank:** Within a single bank in a single channel.
- **Across Banks:** Within a single die (channel) and striped across banks.
- **Across Channels:** Within multiple dies (channels) and striped across one bank in each channel.

E. Impact of Data Striping

If we use an organization that places the entire cache line in the same bank, then a failure of the bank would cause data loss of the entire cache line. To protect stacked DRAM from bank failures or channel failures, we can stripe data across banks or channels. In such a case, each bank/channel would be responsible for only a portion of the data for the cache line, and a correction mechanism (possibly ECC scheme) can be used to fix the sub-line-granularity fault. This organization requires the activation of multiple banks/channels to satisfy each memory request, thereby reducing bank-level parallelism and consuming much higher power by activating multiple banks for every request.

Figure 4 compares the reliability for three data mapping schemes for strong 8-bit symbol based ECC (similar to ChipKill) for different TSV FIT rates (other parameters are described in Section III). System failure is the occurrence of an uncorrectable fault within a seven-year lifetime. Across-Channels configuration provides the highest reliability.

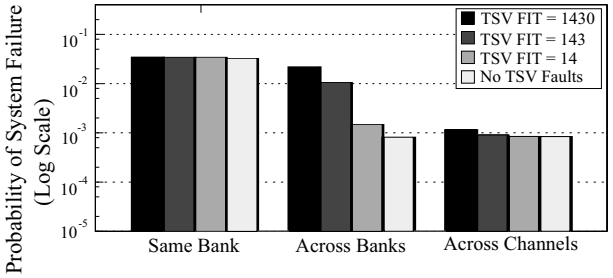

Figure 4. Impact of data striping on Reliability using strong 8-bit symbol based code (similar to Chipkill). Striping data across banks or channels gives higher reliability

Unfortunately, the reliability benefits of Across-Banks and Across-Channels come at a significant price in terms of performance and power. Figure 5 shows that striping data Across-Banks causes a slowdown of approximately 10%, and Across-Channels causes a slowdown of approximately 25%. Furthermore, Across-Channels and Across-Banks consumes 3.8-4.7x more active power than the Same-Bank mapping (Across-Channels takes longer to execute, consuming energy over a longer time, hence the relative reduction in power compared to Across-Banks).

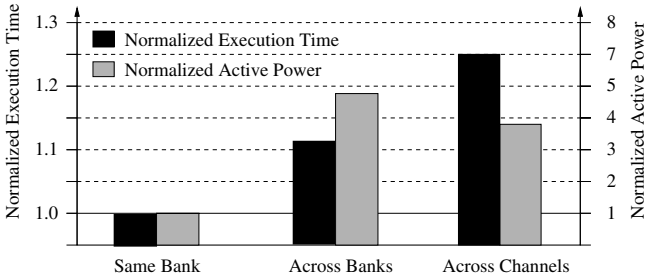

Figure 5. Impact of data striping on Power-Performance. Striping data across banks or channels comes at a significant price in performance (11%-25%) and power (3.8X-4.7X)

Our Goal: The goal of this paper is to obtain high performance and power-efficiency by maintaining the data mapping of a Same-Bank configuration while still making stacked memory robust to large granularity faults. We describe our methodology before describing our solutions.

III. Experimental Methodology

A. Fault Models and Failure Rates

Real-world field data from Sridharan et al. [5] provides DRAM failure rates as Failures In Time (FIT) for 1 Gb DRAM chips. As the technology matures, stacked memory systems will use a much higher density per die (we assume 8 Gb per die, in line with industry projections). We scale the FIT rates based on a proportional increase in the number of common elements determining failure granularity type when going from a 1 Gb to 8 Gb die. An 8 Gb chip will have eight times the number of bits and words in total. Because future standards specify a row buffer size [2, 3] of 2KB, our 8Gb chip with 8 banks would have 64K rows per bank. Sridharn et al. study [5] uses 16K rows in a 1Gb die, we scale the FIT rate for row by 4x for 8Gb dies. Column faults are assumed to originate at the column decoder and scale with the size of the decoder. By our estimates, the number of logic gates in the decoder increases by 1.9x. The bank failure rate is multiplied by 8, since we assume that sub-array size remains roughly constant (to maintain bit line capacitance) [21]. As TSV failure data is not publicly available, we perform a sensitivity study for TSV device FITs. We assume 0.01 to 1 device failures in 7 years (translating to Device FIT of 14 to 1,430) due to TSV faults. Table I shows the failure rates per billion hours (FIT) and the failure sensitivity that we consider for our evaluations.

Table I
Stacked Memory Failure Rates (8Gb Dies)

DRAM Die Failure Mode	Fault Rate (FIT)	
	Transient	Permanent
Single bit	113.6	148.8
Single word	11.2	2.4
Single column	2.6	10.5
Single row	0.8	32.8
Single bank	6.4	80
TSV(Complete Bank/Channel)		
TSV (Address and Data)		Sweep:14 FIT - 1,430 FIT

B. Simulation Infrastructure

Reliability: To evaluate reliability of different schemes, we use a industry-grade fault and repair simulator *FaultSim* [10]. We configure a scrubbing interval of 12 hours. After intervals of 12 hours, correctable transient faults are removed due to the scrubbing mechanism. We conduct the Monte Carlo simulations for $10^5 - 10^6$ trials (more trails for schemes that show lower failure rates, to improve accuracy) for lifetime of 7 years and report an average.

Performance: The baseline configuration is described in Table II. The in-house system simulator uses 8 cores which share an 8 MB LLC. The memory system uses 3D stacks with eight 8 Gb dies for data and one additional die for ECC or metadata in the case of Citadel. Virtual-to-physical translation uses a first-touch policy with a 4KB page size.

Table II
Baseline System Configuration

Processors	
Number of cores	8
Processor clock speed	3.2 GHz
Last-level Cache	
L3 (shared)	8MB, 8-way, 24 cycles
Cache-line size	64Bytes
DRAM 2x8GB 3D stacks	
Memory bus speed	800MHz (DDR3 1.6GHz)
Memory channels	8/Stack
Capacity per channel	1GB
Banks per channel	8
Row-buffer size	2KB
Data TSVs	256/Channel
Addr TSVs	24/Channel
t_{WTR}-t_{CAS}-t_{RCD}-t_{RP}-t_{RAS}	7-9-9-9-36

For our evaluations, we chose all 29 benchmarks from the SPECCPU 2006 [22] suite. We also used memory-intensive benchmarks from the PARSEC [23] suite, such as *black*, *face*, *ferret*, *fluid*, *freq*, *stream* and *swapt*. From the BioBench [24] suite, we used *tigr* and *mummer*. We use a representative slice of 1 billion instructions.

Our evaluations execute the benchmark in rate mode, in which all eight cores execute the same benchmark. We perform timing simulation until all the benchmarks in the workload finish execution, and measure the execution time as the average execution time of all eight cores.

Power: We measure active (read, write, refresh and activation) power using the equations from the Micron Memory System Power Technical Note for 8Gb chip [25, 26]. As per HBM, the refresh interval is set to 32 ms [3, 27].

IV. Citadel: An Overview

We propose *Citadel*, a robust memory architecture that can tolerate both small- and large-granularity faults effectively. Figure 6 shows an overview of Citadel. HBM provisions 64 bits of ECC for every 64 Bytes, possibly in a separate ECC die [3]. Similarly, Citadel provisions each 64B cache line with 64 bits of metadata. However, Citadel uses the ECC die to store different types of metadata information, each geared towards tolerating different types of faults. Each 64B (512b) transaction fetches 40bits of metadata over ECC lanes. The remaining 24 bits are used to provision sparing of faulty blocks. Citadel consists of three component schemes: *TSV-SWAP*, *Tri Dimensional Parity (3DP)* and *Dynamic Dual-Granularity Sparing (DDS)*.

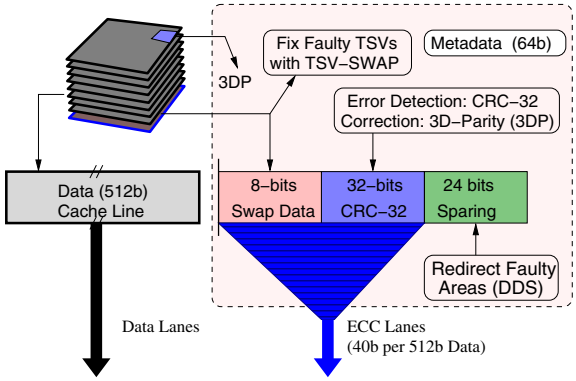

Figure 6. Overview of Citadel

Citadel differentiates faults in memory elements from faults in TSVs. The TSV-SWAP technique of Citadel can tolerate TSV faults by dynamically identifying the faulty TSVs and decommissioning such TSVs. The data of faulty TSVs is replicated in the metadata (up to 8 bits). TSV-SWAP protects against faulty data TSVs as well as faulty address TSVs, which tend to be even more severe in practice. Thus, TSV-Swap provides resilience to TSV faults at runtime, without relying on manufacturer provided spare TSVs.

Citadel relies on CRC to detect data errors. Once an error is detected, it is corrected using the 3DP scheme, which maintains parity in three dimensions: across banks, across rows within one die, and across rows of different dies. 3DP can not only tolerate small-granularity failures such as bit and word failures as well as large-granularity failures such as row and bank failures. 3DP uses one of the data banks to implement bank-level parity (storage overhead of 1.6%).

Citadel employs data sparing to avoid frequent correction of faulty data. This not only prevents the performance overheads of error correction, but also makes the system more robust, as otherwise permanent faults gets accumulated over time. The DDS sparing scheme of Citadel exploits the observation that a bank either has a few small granularity faults (less than 4) or many (more than 1,000) faults; DDS spares at either a row granularity or a bank granularity. DDS uses three out of eight banks of the metadata die for sparing.

When combined, the three techniques of Citadel can tolerate TSV and multi-granularity granularity faults while consuming a storage overhead similar to an ECC DIMM (14% for Citadel versus 12.5% for ECC DIMM) and allowing the data of the cache line to be resident in the same bank. The next sections describe the three techniques in detail.

V. Mitigating TSV Faults with TSV-SWAP

Stacked memory systems use TSVs to connect data, address and command links between the logic die and DRAM dies. Without loss of generality, this section explains the working of TSVs, fault models, and our solution.

A. Background on TSV

The HBM system in this paper consists of 8 channels of 256 Data TSVs (DTSV) with 24 address/command TSVs (ATSV). A memory request presents an address and commands over external address/command links. Internally, TSVs transfer the address and command information for the channel to the corresponding die. For a read request for one cache line, the entire 2KB of data for the row (called a DRAM page) is addressed and brought into the sense amplifiers. From the 2KB (16Kb) page, 64B (512bits) of data are multiplexed and transferred via the TSVs. Because there are only 256 DTSVs, each TSV will transfer data in two DDR cycles. The DRAM row (2KB) contains data for 32 cache lines. Each of these 32 cache lines is multiplexed to the same set of TSVs. Furthermore, all banks within the same die share the TSVs, which means a fault in the TSV causes multi-bank failures.

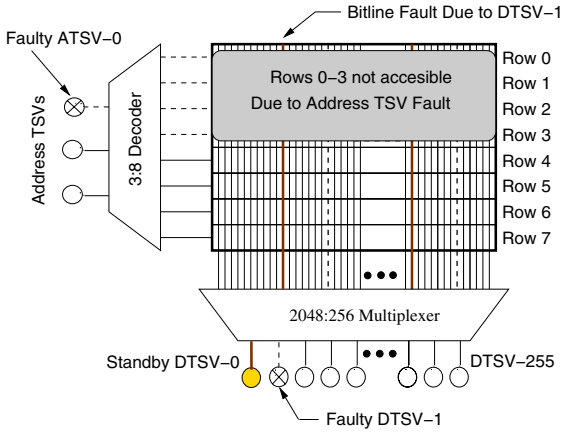

Figure 7. Faults in Data TSV (DTSV) and Address TSV (ATSV). TSV-SWAP creates stand-by TSVs from existing TSVs to tolerate TSV faults (such as DTSV-1 and ATSV-0).

B. Severity of TSV Faults: DTSV vs. ATSV

The vulnerability of the system to TSV faults depends on whether the fault happens in DTSV or ATSV, as shown in Figure 7. Because the burst size for the HBM design is 2, each DTSV fault will cause 2 bits to fail in every cache line. For example, a failure of DTSV-1 will cause bit[1] and

bit[257] of each cache line to fail. Faults in ATSV are even more severe; a single fault can make half of the memory unreachable, because the decoder is unable to address half of the memory space. For example, a failure of ATSV-0 makes half of the rows (Row-0 to Row-3) unreachable.

C. Efficient Runtime TSV Sparing with TSV-SWAP

TSV faults at manufacturing time are typically mitigated by spare TSVs provisioned for enhancing yield [8]. Such spare TSVs may or may not be available to the user to tolerate faulty TSVs that happen at runtime. Our proposal, *TSV-Swap* can mitigate TSV faults at run-time without relying on manufacturer-provided spare TSVs and distinguishes between the severity of faults in address and data TSVs. Instead of relying on spare TSVs, it creates a pool of stand-by TSVs from the available DTSVs, and uses these stand-by TSVs to repair the faulty DTSV and ATSV. If manufacturer-provided spare TSVs are available, then TSV-Swap is still useful as it provides the framework for detecting the faulty TSV at runtime, and repairing the faulty TSV dynamically without data loss. TSV-SWAP consists of three steps and which are described as follows.

1) Creating Stand-by TSVs: TSV-SWAP creates stand-by TSVs by duplicating the data of predefined TSV locations into the 8-bit swap data provided by metadata in Citadel (see Figure 6). Our design designates four TSVs as stand-by TSVs from a pool of 256 DTSV (DTSV-0, DTSV-64, DTSV-128, and DTSV-192). As each DTSV bursts two bits of data for each cache line, 8 bits from each cache line are replicated in the metadata (bit[0], bit[64], ..., bit[448]). The four stand-by TSVs which are created are used to repair any faulty TSVs that occur at runtime.

2) Detecting Faulty TSV: Citadel computes CRC-32 using address and data information. A TSV error will result in an incorrect checksum. To differentiate between TSV faults and data faults, TSV-SWAP employs two additional rows (*row1-fixed* and *row2-fixed*) per die that stores a fixed sequence of data. These rows are at locations where each bit of addresses are the inverse of each other (for example, address *0x0000* and *0xFFFF*). On detecting a CRC mismatch, data from these fixed rows are read and compared against the pre-decided sequence. If there is a mismatch between the compared values, the error is highly likely (but not always) due to a TSV fault. The memory system now invokes the BIST logic which checks for TSV faults.

3) Redirecting Faulty TSV: TSV-SWAP provisions both the DTSV and ATSV with a redirection circuit that can replace a faulty TSV with one of the stand-by TSVs. The redirection circuit is simply a multiplexer and a register. On detecting a TSV fault, the BIST circuitry enables the TSV redirection circuit as a corrective action against the faulty TSV. The BIST circuitry then connects one of the stand-by TSVs to replace the faulty DTSV or ATSV.

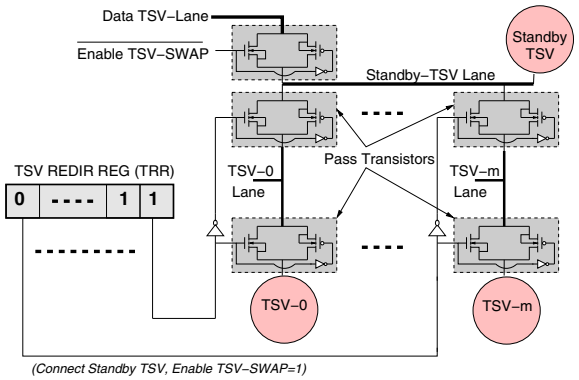

Figure 8. Swap Logic Design for TSV-SWAP. The TRR redirects a faulty TSV to a standy TSV.

TSV-SWAP requires control logic that activates a stand-by TSV for a faulty TSV. Figure 8 shows the design of the swap logic for a pool of data and address TSVs. The *TSV Redir Register (TRR)* stores the sequence after identifying the faulty TSV to be redirected and turns on the required pass transistors. On activating the pass transistor, the TSV lane associated with the faulty TSV gets connected to the stand-by DTSV metal lane. At the same time, *Enable TSV-SWAP* is also set to 1, thereby disconnecting the standby DTSV from its metal lane.[1]

D. Results for TSV-SWAP

We analyze the effectiveness of TSV-Swap at mitigating TSV faults. Unfortunately, the FIT rate data for TSV faults is not available publicly, so for this section, we assume a high TSV fault rate (1430 FIT, corresponding to one TSV-caused die failure every seven years) to assess the effectiveness of TSV-Swap at high TSV fault rate. Figure 9 shows the probability of system failure for the three configurations (No TSV-Swap, With TSV-Swap, and No TSV Faults) for the three data mappings. For all systems, TSV-SWAP achieves a resilience similar to that of not having any TSV faults, even with the assumed high failure rate for TSVs. We conclude that TSV-SWAP is highly effective at mitigating TSV failures. We will assume that all systems employ TSV-Swap for the remainder of the paper, so that we can focus on faults from other sources.

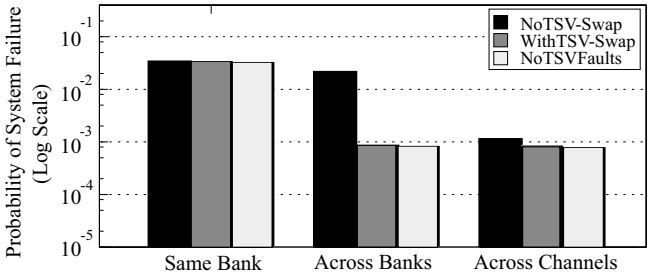

Figure 9. TSV-SWAP is effective at mitigating TSV faults.

[1]On detecting a TSV fault, the TRR gets serially loaded with the information of the faulty TSV using two redundant control TSVs (the chances of both the control TSVs failing is negligible).

VI. Tri Dimensional Parity (3DP)

Even after mitigation of TSV related faults, the stacked memory is still vulnerable to internal DRAM die faults. We want to protect stacked memory not only from small granularity failures (such as bit fault or word fault) but also from large-granularity faults such as column-fault, row-faults or even complete bank failures. The second component of Citadel targets efficient error detection and error correction of data values. Citadel provisions each line with a 32-bit cyclic redundancy code (CRC-32), which is highly effective[2] at detecting data errors [9, 28]. Citadel uses a novel scheme, called *Tri Dimensional Parity (3DP)*, to correct data errors at multiple granularities. In 3DP, even if one dimension encounters two faults, they are highly unlikely to fall into the same block in the other two dimensions. On detecting an error, the memory contents are read and the error gets corrected using parity.[3]

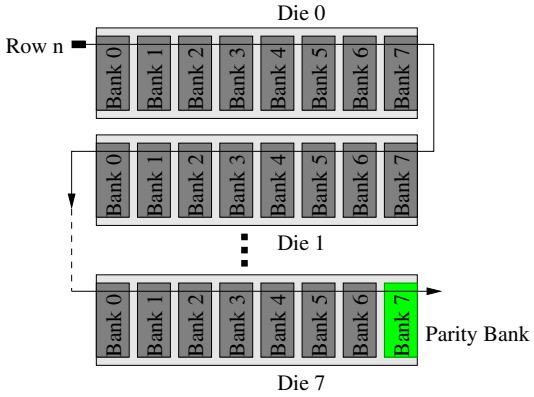

Figure 10. Dimension 1 stripes parity across a single row in every bank for all dies and generates a row in the parity bank

A. Design of Dimension 1

Figure 10 shows the design of Dimension 1. It computes the parity for a row in every bank across dies as specified in equation (1). This requires dedicating a range of single bank addresses as a parity bank for the entire stack (1.6% overhead, for our 8 channel system, with 8 banks for each channel)[4]. A parity bank helps mitigate single-bank faults. However, a one-dimensional parity (1DP) scheme is intolerant of multiple faults. Even if a single-bit failure occurs after a single-bank failure, it results in data loss.

$$ParityBank[row_n] = Die_0.Bank_0[row_n] \oplus Die_0.Bank_1[row_n] \oplus \cdots \oplus Die_7.Bank_6[row_n] \quad (1)$$

[2]The probability of overlapping CRC-32 checksum is $\frac{1}{2^{32}} \approx 10^{-10}$. For false negative, the failed element should have an overlapped CRC-32. The probability that an element fails is less than 10^{-4}. Thus, the effective probability of an overlapping CRC-32 is negligibly small ($\ll 10^{-14}$).

[3]Error correction may take 700 milliseconds, however given that correction is invoked once every few months, this results in negligible performance overheads. We discuss a scheme to avoid persistent errors in the next section.

[4]Parity bank is an abstraction, such a bank can have addresses across multiple physical banks in a stack. This can be done by swapping 2 bits (one lower bank bit and one higher channel bit) while addressing the parity bank. This prevents one physical bank from becoming a bottleneck.

B. Design of Dimensions 2 and 3

Figure 11 shows the design of Dimensions 2 and 3. In Dimension 2, parity is taken across all rows in all banks within a die. Equation (2) shows the computation *Parity Row* in Dimension 2 for Die 0. Because there are 9 dies (including the metadata die), the storage overhead is 9× the size of a DRAM row for each dimension.

$$ParityRowDim2_{Die0} = [Bank_0[row_0] \oplus Bank_0[row_1] \oplus \cdots \oplus Bank_7[row_n]]_{Die0} \quad (2)$$

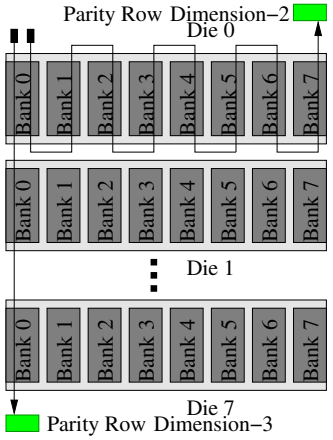

Figure 11. Dimension 2 stripes parity across all row in every bank within a die and generates a parity row. Dimension 3 stripes parity across all row in single bank across dies and generates a parity row.

Dimension 3 computes parity across dies for all rows in a single bank. Equation (3) shows the computation for *Parity Row* in Dimension 3 for Bank 0. Because there are 8 banks per die, the storage overhead of is 8×size of DRAM row. While Dimension 1 is designed to tolerate bank failures, Dimensions 2 and 3 prevent independent row, word and bit failures. When used together, 3DP can correct multiple errors that occur at the same time within a stack.

$$ParityRowDim3_{Bank0} = [Die_0[row_0] \oplus Die_0[row_1] \oplus \cdots \oplus Die_7[row_n]]_{Bank0} \quad (3)$$

C. Reducing Overheads for Parity Update

We avoid the performance overheads of updating the parity for Dimensions 2 and 3 by keeping the parity information on-chip. The size of the row buffer of the stacked DRAM we simulate is 2KB [2, 3]. Thus, maintaining Dimensions 2 and 3 would require a storage overhead of 34 KB (9 rows for Dimension 2 and 8 rows for Dimension 3), which can be kept at the memory controller. Thus, updating the parity for Dimensions 2 and 3 can be done on-chip with negligible timing and power overheads.

The total size of parity for Dimension 1 is equal to 1 Gb (128 MB) which would be impractical to duplicate at the memory controller side. To reduce the parity update

overheads for Dimension 1, we employ parity caching within the on-chip LLC. For Dimension 1, every parity cache line is responsible for 63 data lines from 63 different banks. Thus, we expect accesses to parity lines to have very high temporal locality. Figure 12 shows the operation of a system that implements on-demand parity caching within the LLC for a writeback request to a data line (action ❶).

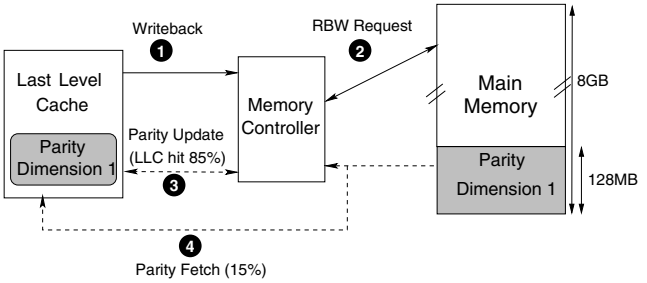

Figure 12. Memory System employing on-demand parity caching for Dimension 1 within the LLC (Figure not to scale)

To update the parity information, we need to get the XOR of the old data and new data of the line for which a writeback request is being made. The memory controller performs such a Read Before Write (RBW) request to obtain the old information of the line (action ❷). The XOR forms a parity update. The memory controller then checks the LLC for the parity line associated for the address for which writeback is being made. In the common case (85% of the time, on average) the parity line is found in the LLC and the parity is updated with the XOR value (action ❸). In the uncommon case that the parity information for Dimension 1 is not found in the LLC, then parity information is fetched from the memory (action ❹), installed in the LLC, and the parity information is updated.

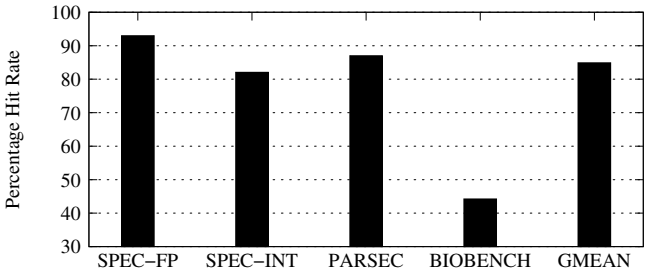

Figure 13. Hit rate for parity caching of Dimension 1

Figure 13 shows the LLC hit-rate for parity update requests. The data is averaged across all workload suites. On average, the hit rate is 85%, showing that parity caching is quite effective. The BIOBENCH workloads mostly perform read operations, with writes sparsely distribute between a large number of writes. Hence, read requests tend to evict parity lines. However, since the frequency of writes for BIOBENCH is less, a low hit rate for parity update results in negligible performance loss.

D. Error Detection and Correction using 3DP

On every read request, 3DP works in two phases. The first phase consists of fast error checking using CRC-32 code. For most requests, this phase will report no errors. However in the rare case of a reported error (once in a few months), the second phase is activated and the whole memory is read. 3DP then isolates the fault(s) using all three dimensions of parity across the stack. If it is a small granularity bit, word or row fault, then dimensions 2 and 3 parity can fix such errors. However, large granularity faults such as column and bank faults are corrected using dimension 1 parity. In the event of simultaneous multi-granularity faults, dimensions 2 and 3 parity help isolate small granularity faults and dimension 1 parity helps isolate the large granularity fault.

E. Results for 3DP

The 3DP scheme allows the memory system to retain the cache line within the same bank, and yet be able to correct bit, word, row, column and bank failures. We compare the resilience, performance, and power of the 3DP scheme to a theoretical scheme that employs an 8-bit symbol-based coding with data striping. For a fair comparison between the two schemes, we assume that TSV-Swap is enabled for both the 8-bit symbol based code and 3DP.

1) Resilience: Figure 14 compares the multi-dimensional parity scheme with a very strong 8-bit symbol- correcting code striped across channels. Enabling only a single dimension of parity (at Bank Level) does not improve resilience against multiple faults that occur concurrently. A single dimensional parity scheme is unable to correct these faults. Increasing the parity dimensions from one to two improves resilience by 100x, because these two levels of parity can isolate most of the faults. By enabling all three dimensions, 3DP achieves a 1,000x improvement in resilience. Furthermore, 3DP achieves 7x stronger resilience than an 8-bit symbol-based ECC because it can handle higher number of multiple concurrent faults.

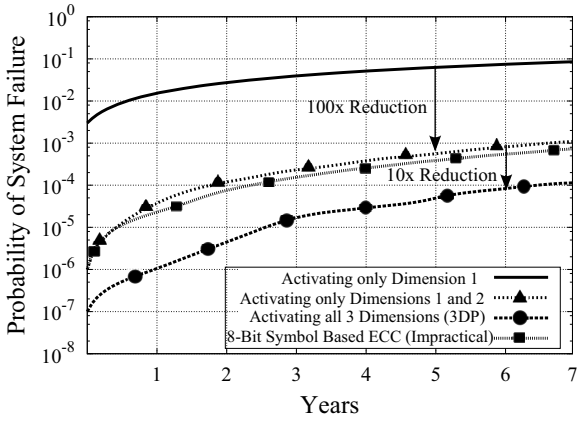

Figure 14. 3DP has 7x more resilience than an 8-bit symbol-based ECC code for tolerating large-granularity failures in stacked memory. 3DP has 10x more resilience than 2DP

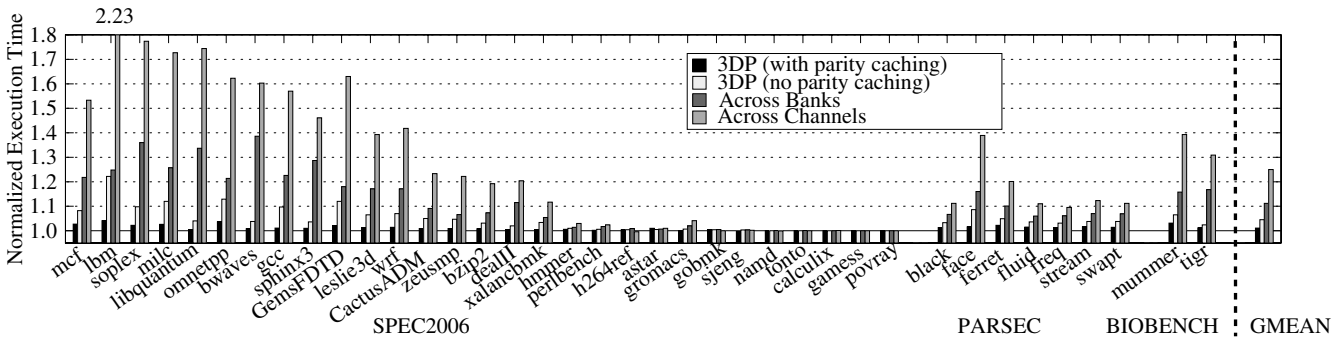

Figure 15. Normalized execution time: 3DP has negligible slow-down, whereas data striping causes 10-25% slow-down.

2) Performance: Figure 15 compares the execution time of 3DP to the organizations that stripe data either across a bank or a channel. The execution time is normalized to a baseline that retains the cache line within the same bank and pays no overhead for error correction. The 3DP scheme with caching has performance within 1% of the baseline, 3DP without caching degrades performance by 4.5%. Thus, parity caching is highly effective at mitigating the performance impact of parity updates. Alternative schemes, that rely on striping the data in different banks or channels, degrade performance by as much as 10% to 25%, on average due to the loss of bank/channel level parallelism. Thus, 3DP not only improves the resilience of stacked memory compared to data striping, but also helps brings the performance impact of fault tolerance to a negligible level.

3) Power: Accessing multiple banks or channels to satisfy every memory request also has the disadvantage that it consumes significantly higher power. Our proposed 3DP design allows Citadel to place the entire cache line in one bank, and thus activate only one bank per read request. This not only reduces the activation power but also improves memory level parallelism, compared to the Across-Bank and Across-Channel configuration. Figure 16 shows the active power for 3DP, Across-Bank, and Across-Channel configuration, normalized to the fault-free baseline that places the cache line in the same bank. On average, 3DP increases active power by only 4%, whereas Across-Bank and Across-Channel configurations increase active power by almost 3X-5X of higher bank/channel activations and row conflicts.

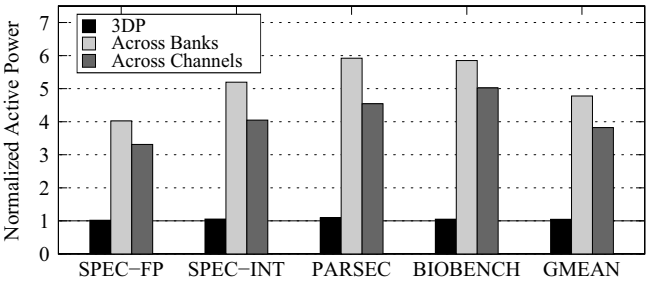

Figure 16. Active power consumption: 3DP has negligible power overheads, whereas data striping incurs 3x-5x power.

VII. Dynamic Dual-granularity Sparing (DDS)

The 3DP scheme performs error correction by recomputing the data based on parity information. However, this can be a time-consuming process (recomputing parity and isolating the fault in each dimension). Fortunately, faults do not occur frequently, so employing a slow correction mechanism is a viable option. However, if the faults are permanent then the correction scheme will be invoked frequently and cause unacceptable performance degradation. Citadel avoids this by using dynamic sparing, whereby a data item once corrected is redirected to an alternate location. The key question in designing a data-sparing scheme is the granularity of sparing. Sparing at row granularity would be storage efficient, however it would be fairly complex to tolerate bank failures, as the redirection structures associated with row sparing would require several tens of thousands of entries. We can implement sparing at a bank granularity, but suffer significant under-utilization of spare area. Thus, uniform sparing is either complex or inefficient. To address this dichotomy, Citadel is provisioned with *Dynamic Dual-granularity Sparing (DDS)*. We present the key observation that motivates DDS.

A. Key Observation: Failures Tend to be Bimodal

Only for the analysis in this section, we will classify all faults that are smaller than or equal to a row fault as causing a row failure. These faults will consume one entry for a row-sparing architecture. A large-granularity fault would consume many entries of row sparing. Figure 17 shows the distribution of the number of rows that are used by a faulty bank, on average. The number of failures show a bimodal distribution. The smaller-granularity faults do not occur in many multiples. In fact, in all our simulations, no more than two rows per bank were affected by a small-granularity fault within a scrubbing interval. However, there are two peaks; one at 5,200 rows (most likely due to sub-arrays) and another at 65K rows (size of a bank). A row-sparing architecture would be not effective at tolerating 65K spare rows for a failed bank, because the sparing associated table would become impractically large to build and search on every access. Therefore, DDS implements two granularities of sparing: either a row or a bank.

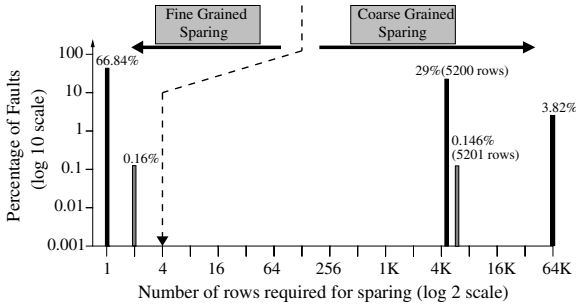

Figure 17. Permanent fault affects either very few (less than 4) rows or large number of ($>$ 1000) rows.

B. Budgeting Spare Rows and Spare Banks

DDS partitions faults into small- and large-granularity faults, then replaces small-granularity faults with rows and large-granularity faults with a bank. Based on the data shown in Figure 17 we deem any bank having more than four faulty rows as a bank failure and spare that bank. Given that a bank can have at most four row failures before the bank gets spared, the total number of spare rows required would be equal to four times the number of banks (for 64 banks there would be 256 spare rows).

The number of spare banks depends on the bank failure rate. Table III shows the distribution of faulty banks for a system that has at least one failed bank (more than four row faults). Even under our conservative definition of bank failure, we need at most two spare banks to handle 99.96% of the systems that have a bank failure, so we employ two spare banks in our design.

Table III
NUM. FAILED BANKS (FOR SYSTEM WITH $\geq$1 BANK FAIL)

Num Faulty Banks	1	2	3+
Probability	66.98%	32.98%	0.04%

C. Design of Dynamic Dual-granularity Sparing

DDS has two components; the spare area and the redirection table. Because we employ two granularities of sparing we have two redirection tables; one at row granularity and the other one at a bank granularity.

1) Spare Area: The metadata die in *Citadel* has 8 banks. TSV and 3DP use 5 banks within the metadata die for storing CRC-32 and TSV-SWAP related information. DDS uses the three remaining banks for sparing. These 3 banks are partitioned into coarse-granularity sparing banks (*spare bank-0* and *spare bank-1*) and a fine granularity bank (*spare bank-2*) that provides space for row-based sparing.

2) Row Remap Table (RRT): DDS uses RRT to associate faulty row addresses with spare row addresses. Each RRT entry contains a valid bit (1), the source row ID (16 bits) and a destination row ID (16 bits). Each fault is tagged with a faulty row address and its corresponding spare address. Because DDS supports at most 4 spare rows for each bank, each bank has 4 entries in RRT. The overhead of RRT for our 8 die (8 banks per die) system is approximately 1 KB and the RRT is stored on-chip. A memory access will check the 4 RRT entries of the given bank for a valid row ID match. On a valid match, the spare row is accessed.

3) Bank Remap Table (BRT): If all four spare rows dedicated to a bank get exhausted, and a new fault appears, then the fault is treated like a large-granularity (bank) failure and coarse-granularity sparing is invoked. The data from the failed bank is repaired and relocated to the spare bank. A two-entry Bank Remap Table (BRT) provides redirection for faulty banks. Each BRT entry contains a valid bit, the ID of the failed bank (6 bit ID), and ID of the spare bank (1 bit spare bank ID, to select one of two spare banks). The BRT is located on chip, and is probed on every memory access for a match, prior to looking up the RRT. On a BRT hit, the spare bank is accessed.

D. Overall Results: Tying it All Together

Figure 18 compares the effectiveness of 3DP with DDS to an 8-bit symbol correcting code. For all systems, we assume that TSV-SWAP is enabled. DDS when applied with 3DP delivers a 700x improvement in resilience compared to the baseline strong 8-bit symbol-based ECC code. DDS removes 99.995% of all transient faults and 99.996% of all the permanent faults with a 12-hour scrubbing interval and thus prevents the accumulation of faults. Therefore, DDS can protect against multiple faults if they occur during different scrub intervals. Overall, these results show that Citadel can provide a reliability improvement of almost three orders of magnitude. It does so without requiring the system to stripe data for a cache line across banks.

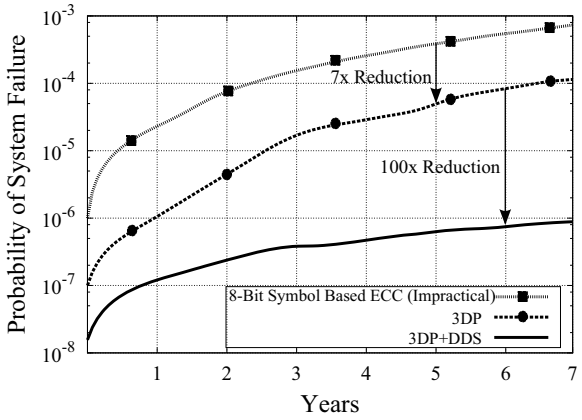

Figure 18. Resilience: 3DP+DDS provides 700x more resilience than symbol-based codes that rely on data striping

E. Overall Storage Overhead of Citadel

Citadel relies on having an extra die for storing metadata for the eight data dies (12.5% overhead). In addition, bank-level parity requires dedicating one of the data bank for storing parity (1.6% overhead, one bank out of 64 banks). For 3DP, we keep parity for Dimensions 2 and 3 on-chip (34 KB overhead), and the redirection tables of DDS incur about 1KB overhead, for a total SRAM overhead of only

35KB. Thus, Citadel provides 700x better reliability while requiring a storage overhead of 14% which is similar to the overhead of ECC DIMM (12.5%).

VIII. Related Work

Memory reliability for emerging memory technologies and existing DRAM systems is an important field. We describe the schemes that are most relevant to our proposal.

A. TSV Reliability using Redundancy

Citadel employs TSV-SWAP to mitigate faulty TSVs. Faulty TSVs can be avoided at manufacturing time using spare TSVs. Several techniques have been proposed for "swapping in" such redundant TSVs to replace faulty TSVs in a 3D die stack [29]. To the best of our knowledge, this paper is the first to address run-time mitigation of TSVs and without relying on manufacturer-provided spare TSVs.

B. Reliability for Stacked DRAM Cache

The work that is most closely related to our work is on reliably architecting stacked DRAM as caches [28]. It uses CRC-32 to detect errors in caches. However, correction is performed simply by disabling clean lines and replicating dirty lines. While such correction can be useful for caches, disabling random locations of lines is an impractical option for main memory. Furthermore, replicating all the data for main memory leads to a capacity loss of 50% and doubles the memory activity. Our work provides low-cost and effective fault tolerance for using stacked DRAM as main memory.

C. Virtual and Flexible Tiered ECC

Yoon et al. [30] proposed *Virtual and Flexible ECC*. Rather than using uniform error correction across the entire memory space, it allows the user to specify stronger levels of ECC for high-priority applications and weaker levels of ECC for low-priority applications. Citadel uses multi-dimensional parity rather than multi-tiered ECC. Citadel is more area-efficient and does not require any support from the OS.

D. Repairing Bit Faults and Uniform Sparing

Efficient memory repair for bit-level faults has been proposed for both SRAM [31][32] and DRAM [33]. However, such techniques are effective only for random bit errors, and become ineffective at tolerating large-granularity faults. Erasure Codes can identify faulty chips to be disabled [34, 35, 36]. However, they can operate only at one granularity. Unlike erasure codes, DDS enables flexible granularity sparing.

E. Prior Work on Parity Based ECC

Parity based schemes such as LOT-ECC and 2D-ECC protect against multi-bit faults [37, 38]. 2D-ECC only protects against small granularity faults (32x32 cells). LOT-ECC has a parity tier, but encounters upto 25% area overhead. On the contrary, 3DP is not a natural extension to 2D-ECC. 3DP stripes data across row buffers and uses a parity bank to protect against large-granularity faults. 3DP is geared towards large-granularity faults and leverages on the physical organization of stacked memories. 3DP has significantly reduced storage(1.6%) and achieves more resilience ($\sim$130X higher) when compared to 2D-ECC.

F. RAID and Stronger BCH Codes

RAID also uses parity for error correction [39]. BCH codes can provide protection for multiple-bit errors (e.g. 6 or more bits) [40][41]. Figure 19 compares the resilience of Citadel with a strong ECC scheme (6EC7ED) and with RAID-5 in a memory system with no TSV faults. Even after discounting for TSV faults, these schemes end up having orders of magnitude higher failure rates than Citadel. 6EC7ED-based codes cannot correct large-granularity faults. A RAID-5 scheme provides 89x improvement in resilience compared to 6EC7ED. Citadel provides 1000x more resilience than a RAID-5 scheme.

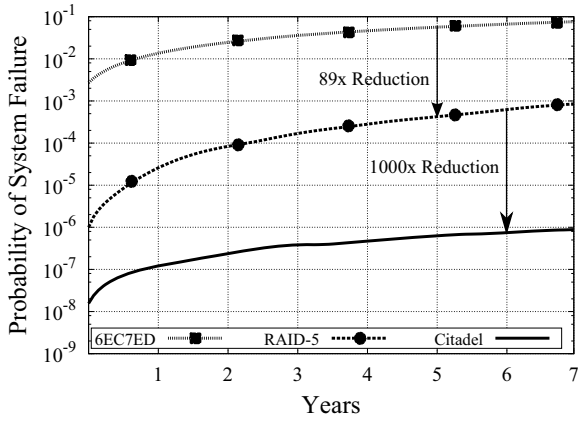

Figure 19. Comparing resilience of Citadel to 6EC7ED code and RAID-5.

IX. Conclusion

memory stacking introduces new multi-bit failure modes, exacerbating the large-granularity faults identified by recent DRAM field studies. Typical approaches to memory fault tolerance tolerate only random-bit failures. Tolerating large-granularity failures (such as tolerating chip failures using ChipKill) typically relies on striping data to multiple chips. Transposing such data striping to stacked memory systems causes significant slowdown and 3-5x power overheads. This paper proposes *Citadel* to tolerate large-granularity faults efficiently, and makes the following contributions:

1) TSV-SWAP, which mitigates TSV faults at run-time, without relying on manufacturer-provided spare TSVs. It remains effective even at high TSV failure rates.

2) Tri-Dimensional Parity (3DP) which can correct a wide variety of multi-granularity faults.
3) Dynamic Dual-granularity sparing (DDS) which can spare faulty data blocks either at a row granularity or at a bank granularity to avoid the accumulation of permanent faults and frequent error correction.

Our evaluations with real-world fault data for DRAM chips shows that combining these three schemes is highly effective for tolerating high rate of TSV failures and memory failures. We show that 3DP improves reliability of stacked memory by 7x, and when combined with DDS by 700x, compared to a symbol-based code that stripes data across banks or channels. Citadel provides high reliability while maintaining high performance and low power, requiring a storage overhead close to ECC DIMMs (14% vs. 12.5%).

ACKNOWLEDGMENT

We thank the anonymous reviewers for their valuable comments and feedback. We also thank Vilas Sridharan for his comments on DRAM scaling and FIT rates, and the members of our research group at Georgia Tech and AMD Research for providing insightful feedback. This work was supported in part by Center for Future Architectures Research (C-FAR), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

REFERENCES

[1] U. Kang *et al.*, "8gb 3d ddr3 dram using through-silicon-via technology," in *ISSCC*, 2009.
[2] H. M. C. Consortium, "Hybrid memory cube specification 1.0," 2013. [Online]. Available: hybridmemorycube.org
[3] J. Standard, "High bandwidth memory (hbm) dram," in *JESD235*, 2013.
[4] M. Dubash, "Not hot swap but 'fail in place'," in *TechWorld*, 2004. [Online]. Available: http://features.techworld.com/storage/960/not-hot-swap-but-fail-in-place/
[5] V. Sridharan and D. Liberty, "A study of dram failures in the field," in *SC-2012*.
[6] *DDR3 ECC Unbuffered DIMM Spec Sheet*, Silicon Power, 2010.
[7] T. J. Dell, "A white paper on the benefits of chipkillcorrect ecc for pc server main memory," IBM, Tech. Rep. 11/19/97, 1997.
[8] A.-C. Hsieh *et al.*, "Tsv redundancy: Architecture and design issues in 3d ic," in *DATE 2010*.
[9] W. Peterson and D. Brown, "Cyclic codes for error detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.
[10] D. Roberts and P. Nair, "Faultsim: A fast, configurable memory-resilience simulator," in *The Memory Forum: ISCA-41*.
[11] B. Schroeder *et al.*, "Dram errors in the wild: a large-scale field study," *SIGMETRICS Perform. Eval. Rev.*
[12] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *Dependable and Secure Computing, IEEE Transactions on*, 2010.
[13] V. Sridharan *et al.*, "Feng shui of supercomputer memory: Positional effects in dram and sram faults," in *SC*, 2013.
[14] J.-H. Yoo *et al.*, "A 32-bank 1 gb self-strobing synchronous dram with 1 gbyte/s bandwidth," *JSSCC*, vol. 31, no. 11, pp. 1635–1644, 1996.
[15] S. Shiratake *et al.*, "A pseudo multi-bank dram with categorized access sequence," in *VLSI*, 1999.
[16] J.-S. Kim *et al.*, "A 1.2v 12.8gb/s 2gb mobile wide-i/o dram with 4x128 i/os using tsv-based stacking," in *ISSCC*, 2011.
[17] J. T. Pawlowski, "Hybrid memory cube (hmc)," in *HOT-CHIPS*, 2011.
[18] T. Hollis, "Modeling and simulation challenges in 3d memories," in *DesignCon*, 2012.
[19] J. Bolaria, "Micron reinvents dram memory," in *Microprocessor Report (MPR)*, 2011.
[20] *Octopus 8-Port DRAM for Die-Stack Applications: TSC100801/2/4*, Tezzaron Semiconductor, 2010.
[21] Y. Kim *et al.*, "A case for exploiting subarray-level parallelism (salp) in dram," in *ISCA-39*.
[22] "Spec cpu2006 benchmark suite," in *Standard Performance Evaluation Corporation*. [Online]. Available: http://www.spec.org/cpu2006/
[23] C. Bienia, "Benchmarking modern multiprocessors," in *Ph.D. Thesis, Princeton University*, 2011.
[24] K. Albayraktaroglu *et al.*, "Biobench: A benchmark suite of bioinformatics applications."
[25] *Calculating Memory System Power for DDR3*, Micron, 2007.
[26] *MT41J512M4:8Gb QuadDie DDR3 SDRAM Rev. A 03/11*, Micron, 2010.
[27] (2011) Jang seok choi in the ddr4 mini workshop. [Online]. Available: http://jedec.org/sites/default/files/JS_Choi_DDR4_miniWorkshop.pdf
[28] J. Sim *et al.*, "Resilient die-stacked dram caches," in *ISCA-40*.
[29] L. Jiang, Q. Xu, and B. Eklow, "On effective tsv repair for 3d-stacked ics," in *DATE-2012*.
[30] D. H. Yoon and M. Erez, "Virtualized and flexible ecc for main memory," in *ASPLOS-15*.
[31] D. Roberts *et al.*, "On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology," in *DSD-10*.
[32] C. Wilkerson and othes, "Trading off cache capacity for reliability to enable low voltage operation," in *ISCA-35*.
[33] P. J. Nair *et al.*, "Archshield: architectural framework for assisting dram scaling by tolerating high error rates," in *ISCA-40*.
[34] J. Nerl *et al.*, "System and method for controlling application of an error correction code (ecc) algorithm in a memory subsystem," Patent US 7 437 651 B2.
[35] D. H. Yoon *et al.*, "Boom: Enabling mobile memory based low-power server dimms," in *ISCA-39*.
[36] J. Nerl *et al.*, "System and method for applying error correction code (ecc) erasure mode and clearing recorded information from a page deallocation table," Patent US 7 313 749 B2.
[37] A. Udipi *et al.*, "Lot-ecc: Localized and tiered reliability mechanisms for commodity memory systems," in *ISCA-39*.
[38] J. Kim *et al.*, "Multi-bit error tolerant caches using two-dimensional error coding," in *MICRO-40*.
[39] A. Thomasian and J. Menon, "Raid5 performance with distributed sparing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 8, no. 6, pp. 640–657, 1997.
[40] S. Li *et al.*, "System implications of memory reliability in exascale computing," in *SC*, 2011.
[41] C. Wilkerson *et al.*, "Reducing cache power with low-cost, multi-bit error-correcting codes," in *ISCA-37*.